

Parameter Synthesis for Cardiac Cell Hybrid Models Using δ -Decisions

Bing Liu¹, Soonho Kong¹, Sicun Gao¹, Paolo Zuliani², and Edmund M. Clarke¹

¹ Computer Science Department, Carnegie Mellon University, USA

² School of Computing Science, Newcastle University, UK

Abstract. A central problem in systems biology is to identify parameter values such that a biological model satisfies some behavioral constraints (*e.g.*, time series). In this paper we focus on parameter synthesis for hybrid (continuous/discrete) models, as many biological systems can possess multiple operational modes with specific continuous dynamics in each mode. These biological systems are naturally modeled as hybrid automata, most often with nonlinear continuous dynamics. However, hybrid automata are notoriously hard to analyze — even simple reachability for hybrid systems with linear differential dynamics is an undecidable problem. In this paper we present a parameter synthesis framework based on δ -complete decision procedures that sidesteps undecidability. We demonstrate our method on two highly nonlinear hybrid models of the cardiac cell action potential. The results show that our parameter synthesis framework is convenient and efficient, and it enabled us to select a suitable model to study and identify crucial parameter ranges related to cardiac disorders.

1 Introduction

Computational modeling and analysis methods are playing a crucial role in understanding the complex dynamics of biological systems [1]. In this paper we address the parameter synthesis problem for hybrid models of biological systems. This problem amounts to finding sets of parameter values for which a model satisfies some precise behavioral constraints, such as time series or reachability properties. We focus on hybrid continuous/discrete models, since one of the key aspects of many biological systems is their differing behavior in various ‘discrete’ modes. For example, it is well-known that the five stages of the cell cycle are driven by the activation of different signaling pathways. Hence, hybrid system models are often used in systems biology (see, *e.g.*, [2,3,4,5,6,7,8,9]).

Hybrid systems combine discrete control computation with continuous-time evolution. The state space of a hybrid system is defined by a finite set of continuous variables and modes. In each mode, the continuous evolution (*flow*) of the system is usually given by the solution of ordinary differential equations (ODEs). At any given time a hybrid system dwells in one of its modes and each variable evolves accordingly to the flow in the mode. Jump conditions control the switch

to another mode, possibly followed by a ‘reset’ of the continuous variables. Thus, the temporal dynamics of a hybrid system is piecewise continuous.

Hybrid models of biological systems often involve many parameters such as rate constants of biochemical reactions, initial conditions, and threshold values in jump conditions. Generally, only a few rate constants will be available or can be measured experimentally — in the latter case the rate constants are obtained by fitting the model to experimental observations. Furthermore, it is also crucial to figure out what initial conditions or jump conditions may lead to an unsafe state of the system, especially when studying hybrid systems used to inform clinical therapy [10]. All such questions fall within the *parameter synthesis* problem, which is extremely difficult for hybrid systems. Even simple reachability questions for hybrid systems with linear (differential) dynamics are undecidable [11]. Therefore, the parameter synthesis problem needs to be relaxed in a sound manner in order to solve it algorithmically — this is the approach we shall follow.

In this paper, we tackle the parameter synthesis problem using δ -complete procedures [12] for deciding first-order formula with arbitrary computable real functions, including solutions of Lipschitz-continuous ODEs [13]. Such procedures may return answers with one-sided δ -bounded errors, thereby overcoming undecidability issues (note that the maximum allowable error δ is an arbitrarily small positive rational). In our approach we describe the set of states of interest as a first-order logic formula and perform bounded model checking [14] to determine reachability of these states. We then adapt an interval constraints propagation based algorithm to explore the parameter space and identify the sets of resulting parameters. We show the applicability of our method by carrying out a thorough case study characterized by highly nonlinear hybrid models. We discriminate two cardiac cell action potential models [15,16] in terms of cell-type specificity and identify parameter ranges for which a cardiac cell may lose excitability. The results show that our method can obtain biological insights that are consistent with experimental observations, and scales to complex systems. In particular, the analysis we carried out in the cardiac case study is difficult to be performed by — if not out of the scope of — state-of-the-art tools [17,18,19,20].

Related Work. A survey of modeling and analysis of biological systems using hybrid models can be found in [21]. Analyzing the properties of biochemical networks using formal verification techniques is being actively pursued by a number of researchers, for which we refer to Brim’s *et al.* recent survey [22]. Of particular interest in our context are parameter synthesis methods for qualitative behavior specifications (*e.g.*, temporal logic formulas). The method introduced in [23] can deal with parameter synthesis for piecewise affine linear systems. For ODEs, Donzé *et al.* [24] explore the parameter space using adaptive sampling and simulation, while Palaniappan *et al.* [25] use a statistical model checking approach. Other techniques perform a sweep of the entire (bounded) parameter space, after it has been discretized [26,27]. Randomized optimization techniques were used for parameter estimation in stochastic hybrid systems [28], while identification techniques for affine systems were used in [29]. The techniques above can handle nonlinear hybrid systems only through sampling and simulation, and so are in-

complete. Our approach is instead δ -complete. It is based on verified numerical integration and constraint programming algorithms, which effectively enable an over-approximation of the system dynamics to be computed. Thus, if a model is found to be unfeasible (i.e. an **unsat** answer is returned, see Section 2 for more details), then this is correct. This behavior better fits with the safety requirements expected by formal verification.

2 δ -Decisions for Hybrid Models

We encode reachability problems of hybrid automata using a first-order language $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ over the reals, which allows the use of a wide range of real functions including nonlinear ODEs. We then use δ -complete decision procedures to find solutions to these formulas to synthesize parameters.

Definition 1 ($\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ -Formulas). *Let \mathcal{F} be a collection of computable real functions. We define:*

$$\begin{aligned} t &:= x \mid f(t(x)), \text{ where } f \in \mathcal{F} \text{ (constants are 0-ary functions);} \\ \varphi &:= t(x) > 0 \mid t(x) \geq 0 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x_i \varphi \mid \forall x_i \varphi. \end{aligned}$$

By computable real function we mean Type 2 computable, which informally requires that a (real) function can be algorithmically evaluated with arbitrary accuracy. Since in general $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ formulas are undecidable, the decision problem needs to be relaxed. In particular, for any $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ formula ϕ and any rational $\delta > 0$ one can obtain a δ -weakening formula ϕ^δ from ϕ by substituting the atoms $t > 0$ with $t > -\delta$ (and similarly for $t \geq 0$). Obviously, ϕ implies ϕ^δ , but not the *vice versa*. Now, the δ -decision problem is deciding correctly whether:

- ϕ is false (**unsat**);
- ϕ^δ is true (**δ -sat**).

If both cases are true, then either decision is correct. In previous work [12,13,30] we presented algorithms (δ -complete decision procedures) for solving δ -decision problems for $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ and for ODEs. These algorithms have been implemented in the dReal toolset [31]. More details on δ -decision problems are in Appendix.

Now we state the encoding for hybrid models. Recall that hybrid automata generalize finite-state automata by permitting continuous-time evolution (or *flow*) in each discrete state (or *mode*). Also, in each mode an *invariant* must be satisfied by the flow, and mode switches are controlled by *jump* conditions.

Definition 2 ($\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ -Representations of Hybrid Automata). *A hybrid automaton in $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ -representation is a tuple*

$$\begin{aligned} H = \langle X, Q, \{\text{flow}_q(\mathbf{x}, \mathbf{y}, t) : q \in Q\}, \{\text{inv}_q(\mathbf{x}) : q \in Q\}, \\ \{\text{jump}_{q \rightarrow q'}(\mathbf{x}, \mathbf{y}) : q, q' \in Q\}, \{\text{init}_q(\mathbf{x}) : q \in Q\} \rangle \end{aligned}$$

where $X \subseteq \mathbb{R}^n$ for some $n \in \mathbb{N}$, $Q = \{q_1, \dots, q_m\}$ is a finite set of modes, and the other components are finite sets of quantifier-free $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ -formulas.

Example 1 (Nonlinear Bouncing Ball). The bouncing ball is a standard hybrid system model. It can be $\mathcal{L}_{\mathbb{R}\mathcal{F}}$ -represented in the following way:

- $X = \mathbb{R}^2$ and $Q = \{q_u, q_d\}$. We use q_u to represent bounce-back mode and q_d the falling mode.
- $\text{flow} = \{\text{flow}_{q_u}(x_0, v_0, x_t, v_t, t), \text{flow}_{q_d}(x_0, v_0, x_t, v_t, t)\}$. We use x to denote the height of the ball and v its velocity. Instead of using time derivatives, we can directly write the flows as integrals over time, using $\mathcal{L}_{\mathbb{R}\mathcal{F}}$ -formulas:
 - $\text{flow}_{q_u}(x_0, v_0, x_t, v_t, t)$ defines the dynamics in the bounce-back phase:

$$(x_t = x_0 + \int_0^t v(s)ds) \wedge (v_t = v_0 + \int_0^t g(1 - \beta v(s)^2)ds)$$

- $\text{flow}_{q_d}(x_0, v_0, x_t, v_t, t)$ defines the dynamics in the falling phase:

$$(x_t = x_0 + \int_0^t v(s)ds) \wedge (v_t = v_0 + \int_0^t g(1 + \beta v(s)^2)ds)$$

where β is a constant. Again, note that the integration terms define Type 2 computable functions.

- $\text{jump} = \{\text{jump}_{q_u \rightarrow q_d}(x, v, x', v'), \text{jump}_{q_d \rightarrow q_u}(x, v, x', v')\}$ where
 - $\text{jump}_{q_u \rightarrow q_d}(x, v, x', v')$ is $(v = 0 \wedge x' = x \wedge v' = v)$.
 - $\text{jump}_{q_d \rightarrow q_u}(x, v, x', v')$ is $(x = 0 \wedge v' = \alpha v \wedge x' = x)$, for some constant α .
- init_{q_d} is $(x = 10 \wedge v = 0)$ and init_{q_u} is \perp .
- inv_{q_d} is $(x \geq 0 \wedge v \geq 0)$ and inv_{q_u} is $(x \geq 0 \wedge v \leq 0)$.

We now show the encoding of bounded reachability, which is used for encoding the parameter synthesis problem. We want to decide whether a given hybrid system reaches a particular region of its state space after following a (bounded) number of discrete transitions, *i.e.*, jumps. First, we need to define auxiliary formulas used for ensuring that a particular mode is picked at a certain step.

Definition 3. Let $Q = \{q_1, \dots, q_m\}$ be a set of modes. For any $q \in Q$, and $i \in \mathbb{N}$, use b_q^i to represent a Boolean variable. We now define

$$\text{enforce}_Q(q, i) = b_q^i \wedge \bigwedge_{p \in Q \setminus \{q\}} \neg b_p^i$$

$$\text{enforce}_Q(q, q', i) = b_q^i \wedge \neg b_{q'}^{i+1} \wedge \bigwedge_{p \in Q \setminus \{q\}} \neg b_p^i \wedge \bigwedge_{p' \in Q \setminus \{q'\}} \neg b_{p'}^{i+1}$$

We omit the subscript Q when the context is clear.

We can now define the following formula that checks whether a *goal* region of the automaton state space is reachable after exactly k discrete transitions. We first state the simpler case of a hybrid system without invariants.

Definition 4 (k -Step Reachability, Invariant-Free Case). Suppose H is an invariant-free hybrid automaton, U a subset of its state space represented by goal, and $M > 0$. The formula $\text{Reach}_{H,U}(k, M)$ is defined as:

$$\begin{aligned}
& \exists^X \mathbf{x}_0 \exists^X \mathbf{x}_0^t \dots \exists^X \mathbf{x}_k \exists^X \mathbf{x}_k^t \exists^{[0,M]} t_0 \dots \exists^{[0,M]} t_k. \\
& \bigvee_{q \in Q} \left(\text{init}_q(\mathbf{x}_0) \wedge \text{flow}_q(\mathbf{x}_0, \mathbf{x}_0^t, t_0) \wedge \text{enforce}(q, 0) \right) \\
& \wedge \bigwedge_{i=0}^{k-1} \left(\bigvee_{q, q' \in Q} \left(\text{jump}_{q \rightarrow q'}(\mathbf{x}_i^t, \mathbf{x}_{i+1}) \wedge \text{enforce}(q, q', i) \right. \right. \\
& \quad \left. \left. \wedge \text{flow}_{q'}(\mathbf{x}_{i+1}, \mathbf{x}_{i+1}^t, t_{i+1}) \wedge \text{enforce}(q', i+1) \right) \right) \\
& \wedge \bigvee_{q \in Q} (\text{goal}_q(\mathbf{x}_k^t) \wedge \text{enforce}(q, k))
\end{aligned}$$

where $\exists^X x$ is a shorthand for $\exists x \in X$.

Intuitively, the trajectories start with some initial state satisfying $\text{init}_q(\mathbf{x}_0)$ for some q . Then, in each step the trajectory follows $\text{flow}_q(\mathbf{x}_i, \mathbf{x}_i^t, t)$ and makes a continuous flow from \mathbf{x}_i to \mathbf{x}_i^t after time t . When the automaton makes a jump from mode q' to q , it resets variables following $\text{jump}_{q' \rightarrow q}(\mathbf{x}_k^t, \mathbf{x}_{k+1})$. The auxiliary enforce formulas ensure that picking $\text{jump}_{q \rightarrow q'}$ in the i -th step enforces picking $\text{flow}_{q'}$ in the $(i+1)$ -th step.

When the invariants are not trivial, we need to ensure that for all the time points along a continuous flow, the invariant condition holds. We need to universally quantify over time, and the encoding is as follows:

Definition 5 (k -Step Reachability, Nontrivial Invariant). Suppose H contains invariants, and U is a subset of the state space represented by goal. The $\mathcal{L}_{\mathbb{R}, \mathcal{F}}$ -formula $\text{Reach}_{H,U}(k, M)$ is defined as:

$$\begin{aligned}
& \exists^X \mathbf{x}_0 \exists^X \mathbf{x}_0^t \dots \exists^X \mathbf{x}_k \exists^X \mathbf{x}_k^t \exists^{[0,M]} t_0 \dots \exists^{[0,M]} t_k. \\
& \bigvee_{q \in Q} \left(\text{init}_q(\mathbf{x}_0) \wedge \text{flow}_q(\mathbf{x}_0, \mathbf{x}_0^t, t_0) \wedge \text{enforce}(q, 0) \right. \\
& \quad \left. \wedge \forall^{[0,t_0]} t \forall^X \mathbf{x} (\text{flow}_q(\mathbf{x}_0, \mathbf{x}, t) \rightarrow \text{inv}_q(\mathbf{x})) \right) \\
& \wedge \bigwedge_{i=0}^{k-1} \left(\bigvee_{q, q' \in Q} \left(\text{jump}_{q \rightarrow q'}(\mathbf{x}_i^t, \mathbf{x}_{i+1}) \wedge \text{flow}_{q'}(\mathbf{x}_{i+1}, \mathbf{x}_{i+1}^t, t_{i+1}) \wedge \text{enforce}(q, q', i) \right. \right. \\
& \quad \left. \left. \wedge \text{enforce}(q', i+1) \wedge \forall^{[0,t_{i+1}]} t \forall^X \mathbf{x} (\text{flow}_{q'}(\mathbf{x}_{i+1}, \mathbf{x}, t) \rightarrow \text{inv}_{q'}(\mathbf{x})) \right) \right) \\
& \wedge \bigvee_{q \in Q} (\text{goal}_q(\mathbf{x}_k^t) \wedge \text{enforce}(q, k)).
\end{aligned}$$

The extra universal quantifier for each continuous flow expresses the requirement that for all the time points between the initial and ending time point ($t \in$

$[0, t_i + 1]$) in a flow, the continuous variables \mathbf{x} must take values that satisfy the invariant conditions $\text{inv}_q(\mathbf{x})$.

Parameter Identification. The parameter identification problem we tackle is basically a k -step reachability question: Is there a parameter combination for which the model reaches the goal region in k steps? If none exists, then the model is *unfeasible*. Otherwise, a witness (*i.e.*, a value for each parameter) is returned. Note that because we ask for δ -decisions, the returned witness might correspond to a *spurious* behavior of the system. The occurrence of such behaviors can be controlled via the precision δ , but in general cannot be eliminated. We have developed the dReach tool (<http://dreal.cs.cmu.edu/dreach.html>) that automatically builds reachability formulas from a hybrid model and a goal description. Such formulas are then verified by our δ -complete solver dReal [31].

3 Case Study

To exemplify different aspects of our parameter synthesis framework, we carried out a case study on models of cardiac cell electrical dynamics. All experiments reported below were done using a machine with an Intel Core i5 3.4GHz processor and 8GB RAM. The precision δ was set to 10^{-4} . The model files are available at <http://www.cs.cmu.edu/~liubing/cmsb14/>.

3.1 Hybrid models of cardiac cells

The heart rhythm is enabled by the electrical activity of cardiac muscle cells, which make up the atria and ventricles. The electrical dynamics of cardiac cells is governed by the organized opening and closing of ion channel gates on the cell membrane. Improper functioning of the cardiac cell ionic channels can cause the cells to lose excitability, which disorders electric wave propagation and leads to cardiac abnormalities such as ventricular *tachycardia* or *fibrillation*. In order to understand the mechanisms of cardiac disorders, hybrid automata models have been recently developed, including the Fenton-Karma (FK) model [15] and the Bueno-Cherry-Fenton (BCF) model [16].

BCF Model. In this model, the change of cells transmembrane potential u , in response to an external stimulus ϵ from neighboring cells, is regulated by a fast ion channel gate v and two slow gates w and s . Figure 1(a) shows the four modes associated with the BCF model. In Mode 1, gates v and w are open and gate s is closed. The transmembrane potassium current causes the decay of u . The cell is resting and waiting for stimulation. We assume an external stimulus ϵ equal to 1 that lasts for 1 millisecond. The stimulation causes u to increase, which may trigger $\text{jump}_{1 \rightarrow 2} : u \geq \theta_o$. When this jump takes place, the system switches to Mode 2 and v starts closing, and the decay rate of u changes. The system will jump to Mode 3 if $u \geq \theta_w$. In Mode 3, w is also closing; u is governed by the potassium current and the calcium current. When $u \geq \theta_v$, Mode 4 can be

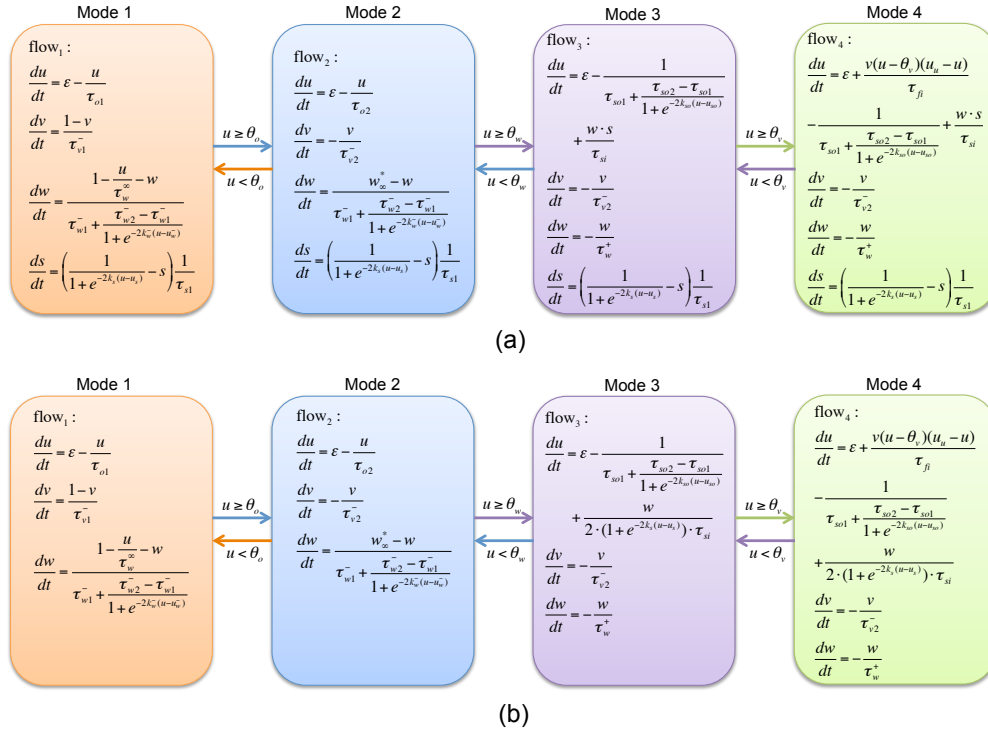


Fig. 1. Hybrid models of cardiac cells. (a) BCF model. (b) FK model.

reached, which signals a successful action potential (AP) initiation. In Mode 4, u reaches its peak due to the fast opening of the sodium channel. The cardiac muscle contracts and u starts decreasing.

FK Model. As shown in Figure 1(b), this model comprises the same four modes and equations of the BCF model, except that the current change induced by gate s is reduced to an explicit term which is integrated in the right-hand side of du/dt . Similarly to the BCF model, an AP can be successfully initiated when Mode 4 is reached.

We specified both the BCF and the FK models using dReach's modeling language. Starting from the state ($u = 0$, $v = 1$, $w = 1$, $s = 0$, $\epsilon \in [0.9, 1.1]$) in Mode 1 (note that the value of s does not matter to FK, which does not contains s), we checked whether Mode 4 is reachable using the parameter values presented in [16]. This was true (*i.e.*, dReach returned δ -sat) for both models (Table 1, Run#1 and Run#2). The simulation of a few witness trajectories are shown in Figure 2 (the stimulus ϵ was reset every 500 milliseconds).

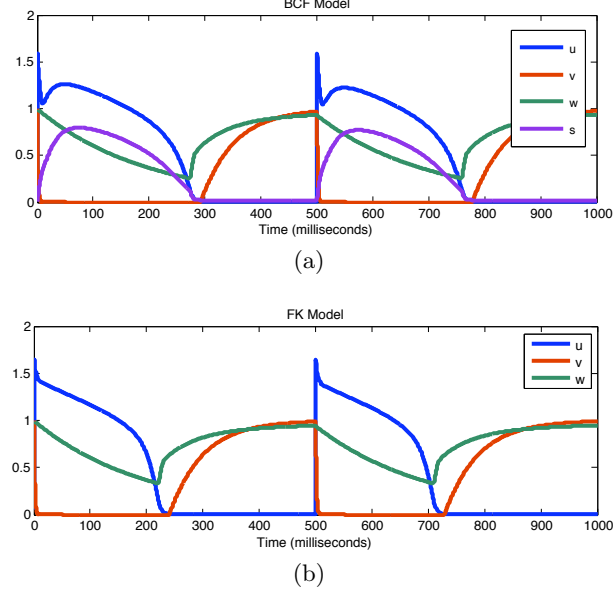


Fig. 2. The simulated witness trajectories of the BCF and the FK models.

3.2 Model falsification

Both the BCF and the FK models were able to reproduce essential characteristics (*e.g.*, steady-state action potential duration) observed in human ventricular cells [15,16]. However, ventricular cells comprise three cell types, which possess different dynamical characteristics. For instance, Figure 3 shows that time courses of APs for epicardial and endocardial human ventricular cells have different morphologies [32]. The important *spike-and-dome* AP morphology can only be observed in epicardial cells but not endocardial cells. Hence, in a model-based study, one needs to identify cell-type-specific parameters to take account into cellular heterogeneity. The feasibility of this task will depend on the model of choice, as for certain models it would be impossible to reproduce a dynamical behavior no matter which parameter values are used. Here we illustrate that such models can be ruled out efficiently using our δ -decision based parameter synthesis framework.

Robustness. We first considered the robustness property of the models. To ensure proper functioning of cardiac cells in noisy environments, an important property of the system is to filter out insignificant stimulation. Thus, we expected to see that AP could not be initiated for small ϵ . Starting from the state ($u = 0$, $v = 1$, $w = 1$, $s = 0$, $\epsilon \in [0.0, 0.25]$) in Mode 1, we checked the reachability of Mode 4. The *unsat* answer was returned by dReach for both the BCF and FK model (Table 1, Run#3 and Run#4), showing that the models are robust to stimulation amplitude.

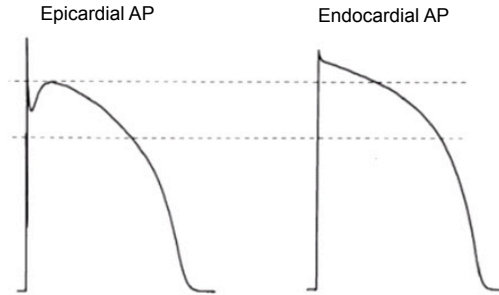


Fig. 3. Different AP morphologies observed in epicardial and endocardial cells [32].

AP morphology. Next we tested whether the models could reproduce the spike-and-dome AP morphology of epicardial cells. We introduced three auxiliary modes (Mode 5, 6 and 7). If $time \geq 1$, the system will jump from Mode 4 to Mode 5, in which ϵ will be reset to 0. The system will jump from Mode 5 to Mode 6 if $time \geq 10$, and will jump from Mode 6 to Mode 7 if $time \geq 30$. In Modes 6 and 7, we enforced invariants $1.0 \leq u \leq 1.15$ and $1.18 \leq u \leq 2.0$, respectively, to depict the spike-and-dome morphology observed experimentally [32]. We then checked reachability of Mode 7, starting from Mode 1 in state $(u = 0, v = 1, w = 1, s = 0, \epsilon \in [0.9, 1.1], \tau_{si} \in [1, 2], u_s \in [0.5, 2])$, where τ_{si} and u_s are two model parameters that govern the dynamics of u and s in Mode 3 and 4 (see Figure 1). The δ -sat answer was returned for BCF (Table 1, Run#5), while *unsat* was returned for FK (Table 1, Run#6), indicating that the FK model cannot reproduce spike-and-dome shapes using reasonable parameter values. Hence, FK is not suitable to study the dynamics of epicardial cells.

We remark that any *unsat* answer is guaranteed to be correct. This effectively means that we proved that the FK model cannot reach Mode 7 for *any* starting state in the rectangle $(u = 0, v = 1, w = 1, s = 0, \epsilon \in [0.9, 1.1], \tau_{si} \in [1, 2], u_s \in [0.5, 2])$. Sampling-based approaches cannot have the same level of certainty, while other approaches cannot handle the complexity of the flows in the model.

3.3 Parameter identification for cardiac disorders

When the system cannot reach Mode 4, the cardiac cell loses excitability, which might lead to tachycardia or fibrillation. Starting with Mode 1, our task was to identify parameter ranges for which the system will never go into Mode 4. In what follows, we focused our study on the BCF model. Grosu *et al.* [33] have tackled this parameter identification problem by linearizing the BCF model into a piecewise-multiaffine system (referred as MHA). With this simplification, parameter ranges could be identified using the Rovergene tool [23]. However, the BCF and MHA models have different sets of parameters. Here we identify disease-related ranges of the *original* BCF parameters. It can be derived from the model equations that τ_{o1} and τ_{o2} govern the dynamics of u in Mode 1 and

Mode 2 respectively, and hence determine whether $\text{jump}_{1 \rightarrow 2}$ and $\text{jump}_{2 \rightarrow 3}$ can be triggered. For τ_{o1} , we performed a binary search in value domain $(0, 0.01]$ to obtain a threshold value θ_{o1} such that Mode 4 is unreachable if $\tau_{o1} < \theta_{o1}$ while Mode 4 is reachable if $\tau_{o1} \geq \theta_{o1}$. The search procedure is illustrated in Algorithm 1. Specifically, we set candidate θ_{o1}^i to be the midpoint of the search domain. We then checked the reachability of Mode 4 with the initial state ($u = 0$, $v = 1$, $w = 1$, $s = 0$, $\theta_{o1} = \theta_{o1}^i$). If $\delta\text{-sat}$ was returned (*e.g.*, Table 1, Run#7), we would recursively check the left-hand half of the search domain; otherwise (*e.g.*, Table 1, Run#8), we would check the other half.

Algorithm 1: Identify parameter threshold value using binary search.

```

1 BinarySearch( $M, v_{min}, v_{max}, \delta$ )
   input : A dReach model  $M$ ; lower and upper bounds of parameter  $v$ :  $v_{min}$ ,
           $v_{max}$ ; precision  $\delta$ 
   output: A threshold value  $\theta_v$ 
2 initialization:  $\theta_v \leftarrow (v_{min} + v_{max})/2$ ;
3 if  $|v_{min} - v_{max}| \leq \delta$  then
4   | return  $\theta_v$  ;
5 else
6   |  $Res \leftarrow \text{dReach}(M, \theta_v, \delta)$  ;
7   | if  $Res = \delta\text{-sat}$  then
8   | | return BinarySearch( $M, v_{min}, \theta_v, \delta$ )
9   | else
10  | | return BinarySearch( $M, \theta_v, v_{max}, \delta$ )
11  | end
12 end

```

In this manner, we identified θ_{o1} to be 0.006, which suggest that when $\tau_{o1} \in (0, 0.006)$, the system will always stay in Mode 1 (Table 1, Run#9). Similarly, we also obtained a threshold value of 0.13 for τ_{o2} , such that Mode 3 cannot be reached when $\tau_{o2} \in (0, 0.13)$ (Table 1, Run#10). Furthermore, whether the system can jump from Mode 3 to Mode 4 depends on the interplay between τ_{so1} and τ_{so2} . For each value τ_{so2}^i of τ_{so2} sampled from domain $[0, 100]$, we performed the binary search in $[0, 5]$ to find the threshold value θ_{so1} such that Mode 4 is unreachable when $\tau_{so1} \in [0, \theta_{so1}]$ and $\tau_{so2} = \tau_{so2}^i$. By linear regression of the obtained values of θ_{so1} , we identified one more condition that Mode 4 is unreachable: $6.2 \cdot \tau_{so1} + \tau_{so2} \geq 9.9$ (*e.g.*, $\tau_{so1} \in [10, 40] \wedge \tau_{so1} \in [0.5, 2]$, see Table 1, Run#11). Taken together, we identified the following disease-related parameter ranges:

$$\tau_{o1} \in (0, 0.006) \vee \tau_{o2} \in (0, 0.13) \vee 6.2 \cdot \tau_{so1} + \tau_{so2} \geq 9.9$$

Figure 4 visualizes these results by showing the simulated trajectories using corresponding parameter values.

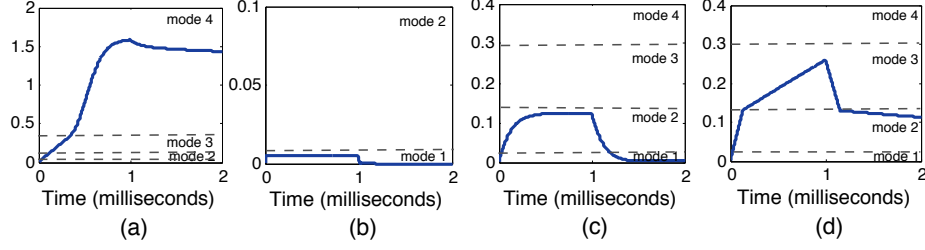


Fig. 4. Simulation results using disease related parameter values. (a) Normal condition (original parameters) (b) $\tau_{o1} = 0.0055$ (c) $\tau_{o2} = 0.125$ (d) $\tau_{so1} = 1.2$, $\tau_{so2} = 1.0$

Run	Model	Initial State	Var	Result	Time
1	BCF	$(u = 0, v = 1, w = 1, s = 0, \epsilon \in [0.9, 1.1])$	53	δ -sat	303
2	FK	$(u = 0, v = 1, w = 1, \epsilon \in [0.9, 1.1])$	53	δ -sat	216
3	BCF	$(u = 0, v = 1, w = 1, s = 0, \epsilon \in [0, 0.25])$	53	unsat	2.09
4	FK	$(u = 0, v = 1, w = 1, \epsilon \in [0, 0.25])$	53	unsat	0.78
5	BCF	$(u = 0, v = 1, w = 1, s = 0, \epsilon \in [0.9, 1.1])$	89	δ -sat	7,904
6	FK	$(u = 0, v = 1, w = 1, \epsilon \in [0.9, 1.1], \tau_{si} \in [1, 2], u_s \in [0.5, 2])$	119	unsat	0.06
7	BCF	$(u = 0, v = 1, w = 1, s = 0, \tau_{o1} = 30.02)$	53	δ -sat	0.89
8	BCF	$(u = 0, v = 1, w = 1, s = 0, \tau_{o1} = 0.0055)$	53	unsat	1.33
9	BCF	$(u = 0, v = 1, w = 1, s = 0, \tau_{o1} \in (0.0, 0.006))$	62	unsat	0.76
10	BCF	$(u = 0, v = 1, w = 1, s = 0, \tau_{o2} \in (0.0, 0.13))$	62	unsat	0.32
11	BCF	$(u = 0, v = 1, w = 1, s = 0, \tau_{so1} \in [10, 40], \tau_{so1} \in [0.5, 2])$	71	unsat	0.11

Table 1. Experimental results. Var = number of variables in the unrolled formula, Result = bounded model checking result, Time = CPU time (s), $\delta = 10^{-4}$.

4 Conclusion

We have presented a framework using δ -complete decision procedures for the parameter identification of hybrid biological systems. We have used δ -satisfiable formulas to describe parameterized hybrid automata and to encode parameter synthesis problems. We have employed δ -decision procedures to perform bounded model checking, and developed an algorithm to obtain the resulting parameters. Our verified numerical integration and constraint programming algorithms effectively compute an over-approximation of the system dynamics. An **unsat** answer can always be trusted, while a **δ -sat** answer might be due to the over-approximation (see Section 2 for more details). We chose this behavior as it better fits with the safety requirements expected by formal verification. We have demonstrated the applicability of our method on a highly nonlinear hybrid model of a cardiac cell that are difficult to analyze with other verification tools. We have successfully ruled out a model candidate which did not fit experimental observations, and we have identified critical parameter ranges that can induce cardiac disorders.

It is worth noting that our method can be applied to ODE based models with discrete events, which are special forms of hybrid automata. Such models are often specified using the Systems Biology Markup Language (SBML) and archived in the BioModels database [34]. Currently, we are currently developing an SBML-to-dReal translator to facilitate the δ -decision based analysis of SBML models. Further, our method also has the potential to be applied to other model formalisms such as hybrid functional Petri nets [35] and the formalisms realized in Ptolemy [36]. We plan to explore this in future work. Another interesting direction is applying our method for parameter estimation from experimental data. By properly encoding the noisy wet-lab experimental data using logic formulas, bounded model checking can be utilized to find the unknown parameter values. In this respect, the specification logic used in [25] promises to offer helpful pointers.

Acknowledgements

This work has been partially supported by award N00014-13-1-0090 of the US Office of Naval Research and award CNS0926181 of the National Science foundation (NSF).

References

1. Liu, B., Thiagarajan, P.: Modeling and analysis of biopathways dynamics. *Journal of Bioinformatics and Computational Biology* **10**(4) (2012) 1231001
2. Chen, K.C., Calzone, L., Csikasz-Nagy, A., Cross, F.R., Novak, B., Tyson, J.J.: Integrative analysis of cell cycle control in budding yeast. *Mol. Biol. Cell* **15**(8) (2004) 3841–3862
3. Ghosh, R., Tomlin, C.: Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-notch protein signalling. *IET Syst. Biol.* **1**(1) (2004) 170–183
4. Hu, J., Wu, W.C., Sastry, S.: Modeling subtilin production in *Bacillus subtilis* using stochastic hybrid systems. In: HSCC. Volume 2993 of LNCS. (2004) 417–431
5. Ye, P., Entcheva, E., Smolka, S., Grosu, R.: Modelling excitable cells using cycle-linear hybrid automata. *IET Syst. Biol.* **2**(1) (2008) 24–32
6. Aihara, K., Suzuki, H.: Theory of hybrid dynamical systems and its applications to biological and medical systems. *Phil. Trans. R. Soc. A* **368**(1930) (2010) 4893–4914
7. Antonioti, M., Mishra, B., Piazza, C., Policriti, A., Simeoni, M.: Modeling cellular behavior with hybrid automata: Bisimulation and collapsing. In: CMSB. Volume 8130 of LNCS. (2003) 57–74
8. Lincoln, P., Tiwari, A.: Symbolic systems biology: Hybrid modeling and analysis of biological networks. In: HSCC. Springer (2004) 660–672
9. Baldazzi, V., Monteiro, P.T., Page, M., Ropers, D., Geiselmann, J., De Jong, H.: Qualitative analysis of genetic regulatory networks in bacteria. In: Understanding the Dynamics of Biological Systems. Springer (2011) 111–130
10. Tanaka, G., Hirata, Y., Goldenberg, S.L., Bruchovsky, N., Aihara, K.: Mathematical modelling of prostate cancer growth and its application to hormone therapy. *Phil. Trans. R. Soc. A* **368** (2010) 5029–5044

11. Henzinger, T.A.: The theory of hybrid automata. In: LICS. (1996) 278–292
12. Gao, S., Avigad, J., Clarke, E.M.: Delta-complete decision procedures for satisfiability over the reals. In: IJCAR. (2012) 286–300
13. Gao, S., Avigad, J., Clarke, E.M.: Delta-decidability over the reals. In: LICS. (2012) 305–314
14. Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic model checking without BDDs. In: TACAS. Volume 1579 of LNCS. (1999) 193–207
15. Fenton, F., Karma, A.: Vortex dynamics in 3D continuous myocardium with fiber rotation: filament instability and fibrillation. *Chaos* **8** (1998) 20–47
16. Bueno-Orovio, A., Cherry, E.M., Fenton, F.H.: Minimal model for human ventricular action potentials in tissue. *J. Theor. Biol.* **253** (2008) 544–560
17. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: Probabilistic symbolic model checker. In: TOOLS. Volume 2324 of LNCS. (2002) 200–204
18. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: CAV. Volume 6174 of LNCS. (2010) 167–170
19. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In: TACAS. Volume 6605 of LNCS. (2011) 254–257
20. Chabrier-Rivier, N., Fages, F., Soliman, S.: The biochemical abstract machine BIOCHAM. In: CMSB. Volume 3082 of LNCS. (2005) 172–191
21. Bortolussi, L., Policriti, A.: Hybrid systems and biology. In: FMCSB. Volume 5016 of LNCS. (2008) 424–448
22. Brim, L., Česka, M., Šafránek, D.: Model checking of biological systems. In: SFM. Volume 7938 of LNCS. (2013) 63–112
23. Batt, G., Belta, C., Weiss, R.: Temporal logic analysis of gene networks under parameter uncertainty. *IEEE T. Automat. Contr.* **53** (2008) 215–229
24. Donzé, A., Clermont, G., Langmead, C.J.: Parameter synthesis in nonlinear dynamical systems: Application to systems biology. *J. Comput. Biol.* **17**(3) (2010) 325–336
25. Palaniappan, S.K., Gyori, B.M., Liu, B., Hsu, D., Thiagarajan, P.S.: Statistical model checking based calibration and analysis of bio-pathway models. In: CMSB. Volume 8130 of LNCS. (2013) 120–134
26. Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine learning biochemical networks from temporal logic properties. In: T. Comp. Sys. Biology. Volume 4220 of LNCS. (2006) 68–94
27. Donaldson, R., Gilbert, D.: A model checking approach to the parameter estimation of biochemical pathways. In: CMSB. Volume 5307 of LNCS. (2008) 269–287
28. Koutroumpas, K., Cinquemani, E., Kouretas, P., Lygeros, J.: Parameter identification for stochastic hybrid systems using randomized optimization: A case study on subtilin production by *Bacillus subtilis*. *Nonlinear Anal.-Hybrid Syst.* **2** (2008) 786–802
29. Cinquemani, E., Porreca, R., Ferrari-Trecate, G., Lygeros, J.: Subtilin production by *Bacillus subtilis*: Stochastic hybrid models and parameter identification. *IEEE Trans. Automat. Contr.* **53** (2008) 38–50
30. Gao, S., Kong, S., Clarke, E.M.: Satisfiability modulo ODEs. In: FMCAD. (2013) 105–112
31. Gao, S., Kong, S., Clarke, E.M.: dReal: An SMT solver for nonlinear theories of reals. In: CADE. (2013) 208–214
32. Nabauer, M., Beuckelmann, D.J., Uberfuhr, P., Steinbeck, G.: Regional differences in current density and rate-dependent properties of the transient outward current

- in subepicardial and subendocardial myocytes of human left ventricle. *Circulation* **93** (1996) 169–177
33. Grosu, R., Batt, G., Fenton, F.H., Gilmm, J., Guernic, C.L., Smolka, S.A., Bartocci, E.: From cardiac cells to genetic regulatory networks. In: CAV. Volume 8130 of LNCS. (2011) 396–411
34. Li, C., Donizelli, M., Rodriguez, N., Dharuri, H., Endler, L., Chelliah, V., Li, L., He, E., Henry, A., Stefan, M.I., Snoep, J.L., Hucka, M., Novere, N.L., Laibe, C.: BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Sys. Biol.* **4** (2010) 92
35. Matsuno, H., Tanaka, Y., Aoshima, H., Doi, A., Matsui, M., Miyano, S.: Biopathways representation and simulation on hybrid functional petri net. In *Silico Biol.* **3**(3) (2003) 389–404
36. Ptolemaeus, C., ed.: *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org (2014)
37. Weihrauch, K.: *Computable Analysis: An Introduction*. Springer (2000)

Appendix: $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ -Formulas and δ -Decidability

We will use a logical language over the real numbers that allows arbitrary *computable real functions* [37]. We write $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ to represent this language. Intuitively, a real function is computable if it can be numerically simulated up to an arbitrary precision. For the purpose of this paper, it suffices to know that almost all the functions that are needed in describing hybrid systems are Type 2 computable, such as polynomials, exponentiation, logarithm, trigonometric functions, and solution functions of Lipschitz-continuous ordinary differential equations.

More formally, $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}} = \langle \mathcal{F}, > \rangle$ represents the first-order signature over the reals with the set \mathcal{F} of computable real functions, which contains all the functions mentioned above. Note that constants are included as 0-ary functions. $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ -formulas are evaluated in the standard way over the structure $\mathbb{R}_{\mathcal{F}} = \langle \mathbb{R}, \mathcal{F}^{\mathbb{R}}, >^{\mathbb{R}} \rangle$. It is not hard to see that we can put any $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ -formula in a normal form, such that its atomic formulas are of the form $t(x_1, \dots, x_n) > 0$ or $t(x_1, \dots, x_n) \geq 0$, with $t(x_1, \dots, x_n)$ composed of functions in \mathcal{F} . To avoid extra preprocessing of formulas, we can explicitly define $\mathcal{L}_{\mathcal{F}}$ -formulas as follows.

Definition 6 ($\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ -Formulas). *Let \mathcal{F} be a collection of computable real functions. We define:*

$$\begin{aligned} t &:= x \mid f(t(\mathbf{x})), \text{ where } f \in \mathcal{F} \text{ (constants are 0-ary functions);} \\ \varphi &:= t(\mathbf{x}) > 0 \mid t(\mathbf{x}) \geq 0 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x_i \varphi \mid \forall x_i \varphi. \end{aligned}$$

In this setting $\neg\varphi$ is regarded as an inductively defined operation which replaces atomic formulas $t > 0$ with $-t \geq 0$, atomic formulas $t \geq 0$ with $-t > 0$, switches \wedge and \vee , and switches \forall and \exists .

Definition 7 (Bounded $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$ -Sentences). *We define the bounded quantifiers $\exists^{[u,v]}$ and $\forall^{[u,v]}$ as $\exists^{[u,v]}x.\varphi =_{df} \exists x.(u \leq x \wedge x \leq v \wedge \varphi)$ and $\forall^{[u,v]}x.\varphi =_{df}$*

$\forall x.((u \leq x \wedge x \leq v) \rightarrow \varphi)$ where u and v denote $\mathcal{L}_{\mathbb{R}_F}$ terms, whose variables only contain free variables in φ excluding x . A bounded $\mathcal{L}_{\mathbb{R}_F}$ -sentence is

$$Q_1^{[u_1, v_1]} x_1 \dots Q_n^{[u_n, v_n]} x_n \psi(x_1, \dots, x_n),$$

where $Q_i^{[u_i, v_i]}$ are bounded quantifiers, and $\psi(x_1, \dots, x_n)$ is quantifier-free.

Definition 8 (δ -Variants). Let $\delta \in \mathbb{Q}^+ \cup \{0\}$, and φ an $\mathcal{L}_{\mathbb{R}_F}$ -formula

$$\varphi : Q_1^{I_1} x_1 \dots Q_n^{I_n} x_n \psi[t_i(\mathbf{x}, \mathbf{y}) > 0; t_j(\mathbf{x}, \mathbf{y}) \geq 0],$$

where $i \in \{1, \dots, k\}$ and $j \in \{k+1, \dots, m\}$. The δ -weakening φ^δ of φ is defined as the result of replacing each atom $t_i > 0$ by $t_i > -\delta$ and $t_j \geq 0$ by $t_j \geq -\delta$:

$$\varphi^\delta : Q_1^{I_1} x_1 \dots Q_n^{I_n} x_n \psi[t_i(\mathbf{x}, \mathbf{y}) > -\delta; t_j(\mathbf{x}, \mathbf{y}) \geq -\delta].$$

It is clear that $\varphi \rightarrow \varphi^\delta$ (see [13]).

In [12], we have proved that the following δ -decision problem is decidable, which is the basis of our framework.

Theorem 1 (δ -Decidability [12]). Let $\delta \in \mathbb{Q}^+$ be arbitrary. There is an algorithm which, given any bounded $\mathcal{L}_{\mathbb{R}_F}$ -sentence φ , correctly returns one of the following two answers:

- δ -True: φ^δ is true.
- False: φ is false.

When the two cases overlap, either answer is correct.

The following theorem states the (relative) complexity of the δ -decision problem. A bounded Σ_n sentence is a bounded $\mathcal{L}_{\mathbb{R}_F}$ -sentence with n alternating quantifier blocks starting with \exists .

Theorem 2 (Complexity [13]). Let S be a class of $\mathcal{L}_{\mathbb{R}_F}$ -sentences, such that for any φ in S , the terms in φ are in Type 2 complexity class \mathcal{C} . Then, for any $\delta \in \mathbb{Q}^+$, the δ -decision problem for bounded Σ_n -sentences in S is in $(\Sigma_n^P)^\mathcal{C}$.

Basically, the theorem says that increasing the number of quantifier alternations will in general increase the complexity of the problem, unless $P = NP$ (recall that $\Sigma_0^P = P$ and $\Sigma_1^P = NP$). This result can be specialized for specific families of functions. For example, with polynomially-computable functions, the δ -decision problem for bounded Σ_n -sentences is (Σ_n^P) -complete. For more details and results we again point the interested reader to [13].